




Try the *new* Portal design
Give us your opinion after using it.

Search Results

Search Results for: `[class loader<AND>((class loading<AND>((class load))))]`
Found 60 of 122,783 searched.

Search within Results

 [> Advanced Search](#)
[> Search Help/Tips](#)

Sort by: [Title](#) [Publication](#) [Publication Date](#) [Score](#)  [Binder](#)

Results 1 - 20 of 60 [short listing](#)



1

2


3

4



1 [Dynamic class loading in the Java virtual machine](#)

98%


 Sheng Liang , Gilad Bracha

ACM SIGPLAN Notices , Proceedings of the conference on Object-oriented programming, systems, languages, and applications October 1998
Volume 33 Issue 10

Class loaders are a powerful mechanism for dynamically loading software components on the Java platform. They are unusual in supporting all of the following features: *laziness*, *type-safe linkage*, *user-defined extensibility*, and *multiple communicating namespaces*. We present the notion of class loaders and demonstrate some of their interesting uses. In addition, we discuss how to maintain type safety in the presence of user-defined dynamic class loading.

2 [Formalizing the safety of Java, the Java virtual machine, and Java card](#)

96%


 Pieter H. Hartel , Luc Moreau

ACM Computing Surveys (CSUR) December 2001
Volume 33 Issue 4

We review the existing literature on Java safety, emphasizing formal approaches, and the impact of Java safety on small footprint devices such as smartcards. The conclusion is that although a lot of good work has been done, a more concerted effort is needed to build a coherent set of machine-readable formal models of the whole of Java and its implementation. This is a formidable task but we believe it is essential to build trust in Java safety, and thence to achieve ITSEC level 6 or Common Crite ...

3 [Multitasking without compromise: a virtual machine evolution](#)

89%

 Grzegorz Czajkowski , Laurent Daynés

ACM SIGPLAN Notices , Proceedings of the OOPSLA '01 conference on Object Oriented Programming Systems Languages and Applications October 2001

The multitasking virtual machine (called from now on simply MVM) is a modification of the Java virtual machine. It enables safe, secure, and scalable multitasking. Safety is achieved by strict isolation of application from one another. Resource control augment security by preventing some denial-of-service attacks. Improved scalability results from an aggressive application of the main design principle of MVM: share as much of the runtime as possible among applications and replicate everything el ...

4 A framework for interprocedural optimization in the presence of dynamic class loading 87%

4 Vugranam C. Sreedhar , Michael Burke , Jong-Deok Choi

ACM SIGPLAN Notices , Proceedings of the ACM SIGPLAN 2000 conference on Programming language design and implementation May 2000

Volume 35 Issue 5

Dynamic class loading during program execution in the Java Programming Language is an impediment for generating code that is as efficient as code generated using static whole-program analysis and optimization. Whole-program analysis and optimization is possible for languages, such as C++, that do not allow new classes and/or methods to be loaded during program execution. One solution for performing whole-program analysis and avoiding incorrect execution after a new class is loaded is to in ...

5 Implementing jalapeño in Java 87%

4 Bowen Alpern , C. R. Attanasio , Anthony Cocchi , Derek Lieber , Stephen Smith , Ton Ngo , John J. Barton , Susan Flynn Hummel , Janice C. Sheperd , Mark Mergen

ACM SIGPLAN Notices , Proceedings of the 1999 ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications October 1999

Volume 34 Issue 10

Jalapeño is a virtual machine for Java™ servers written in Java. A running Java program involves four layers of functionality: the user code, the virtual-machine, the operating system, and the hardware. By drawing the Java / non-Java boundary below the virtual machine rather than above it, Jalapeño reduces the boundary-crossing overhead and opens up more opportunities for optimization. To get Jalapeño started, a boot image of a ...

6 A formal specification of Java class loading 85%

4 Zhenyu Qian , Allen Goldberg , Alessandro Coglio

ACM SIGPLAN Notices , Proceedings of the conference on Object-oriented programming, systems, languages, and applications October 2000

Volume 35 Issue 10

The Java Virtual Machine (JVM) has a novel and powerful mechanism to support lazy, dynamic class loading according to user-definable policies. Class loading directly impacts type safety, on which the security of Java applications is based. Conceptual bugs in the loading mechanism were found in earlier versions of the JVM that lead to type violations. A deeper understanding of the class loading mechanism, through such means as formal analysis, will improve our confidence that no additional bugs a ...

7 A specification of Java loading and bytecode verification 85%

4 Allen Goldberg

Proceedings of the 5th ACM conference on Computer and communications security November 1998

- 8 Intermediate representation engineering: Annotating Java libraries in support of whole-program optimization 84%
Michael Thies
Proceedings of the inaugural conference on the Principles and Practice of programming, 2002 and Proceedings of the second workshop on Intermediate representation engineering for virtual machines, 2002 June 2002
Optimizing just-in-time compilation of Java programs depends on information gained from state-of-the-art program analysis techniques. To avoid extensive analysis at program execution time, analysis results for the available parts of a program can be precomputed at compile time and then combined at runtime. Program analyses operate on isolated program modules like libraries and annotate them with information that can be post-processed efficiently. We have applied this approach to several concrete ...
- 9 Incomunicado: efficient communication for isolates 84%
Krzysztof Palacz , Jan Vitek , Grzegorz Czajkowski , Laurent Daynas
ACM SIGPLAN Notices , Proceedings of the 17th ACM conference on Object-oriented programming, systems, languages, and applications November 2002
Volume 37 Issue 11
Executing computations in a single instance of safe language virtual machine can improve performance and overall platform scalability. It also poses various challenges. One of them is providing a fast inter-application communication mechanism. In addition to being efficient, such a mechanism should not violate any functional and non-functional properties of its environment, and should also support enforcement of application-specific security policies. This paper explores the design and implement ...
- 10 Type-Safe linking with recursive DLLs and shared libraries 84%
Dominic Duggan
ACM Transactions on Programming Languages and Systems (TOPLAS) November 2002
Volume 24 Issue 6
Component-based programming is an increasingly prevalent theme in software development, motivating the need for expressive and safe module interconnection languages. Dynamic linking is an important requirement for module interconnection languages, as exemplified by dynamic link libraries (DLLs) and class loaders in operating systems and Java, respectively. A semantics is given for a type-safe module interconnection language that supports shared libraries and dynamic linking, as well as circular ...
- 11 Application isolation in the Java Virtual Machine 84%
Grzegorz Czajkowski
ACM SIGPLAN Notices , Proceedings of the conference on Object-oriented programming, systems, languages, and applications October 2000
Volume 35 Issue 10
To date, systems offering multitasking for the Java[®] programming language either use one process or one class loader for each application. Both approaches are unsatisfactory. Using operating system processes is expensive, scales poorly and does not fully exploit the protection features inherent in a safe language. Class loaders replicate application code, obscure the type system, and non-uniformly treat 'trusted' and 'untrusted' classes, which leads to subtle, but nevertheless, potenti ...

12 Using complete system simulation to characterize SPECjvm98 benchmarks

84%

4 Tao Li , Lizy Kurian John , Vijaykrishnan Narayanan , Anand Sivasubramaniam , Jyotsna Sabarinathan , Anupama Murthy

Proceedings of the 14th international conference on Supercomputing May 2000

Complete system simulation to understand the influence of architecture and operating systems on application execution has been identified to be crucial for systems design. While there have been previous attempts at understanding the architectural impact of Java programs, there has been no prior work investigating the operating system (kernel) activity during their executions. This problem is particularly interesting in the context of Java since it is not only the application that can invoke ...

13 Executable JVM model for analytical reasoning: a study

82%

4 Hanbing Liu , J Strother Moore

Proceedings of the 2003 workshop on Interpreters, Virtual Machines and Emulators June 2003

To study the properties of the Java Virtual Machine(JVM) and Java programs, our research group has produced a series of JVM models written in a functional subset of Common Lisp. In this paper, we present our most complete JVM model from this series, namely, M6, which is derived from a careful study of the J2ME KVM [16] implementation. On the one hand, our JVM model is a conventional machine emulator. M6 models accurately almost all aspects of the KVM implementation, including the dynamic class lo ...

14 Reflections on remote reflection

82%

4 Michael Richmond , James Noble

Australian Computer Science Communications , Proceedings of the 24th Australasian conference on Computer science January 2001
Volume 23 Issue 1

The Java programming language provides both reflection and remote method invocation: reflection allows a program to inspect itself and its runtime environment, remote method invocation allows methods to be invoked transparently across a network. Unfortunately, the standard Java implementations of reflection and remote method invocation are incompatible: programmers cannot reflect on a remote application. We describe how Java systems can be extended to support **Remote Reflection** transparentl ...

15 Compressing Java class files

82%

4 William Pugh

ACM SIGPLAN Notices , Proceedings of the ACM SIGPLAN 1999 conference on Programming language design and implementation May 1999
Volume 34 Issue 5

Java class files are often distributed as jar files, which are collections of individually compressed class files (and possibly other files). Jar files are typically about 1/2 the size of the original class files due to compression. I have developed a wire-code format for collections of Java class files. This format is typically 1/2 to 1/5 of the size of the corresponding compressed jar file (1/4 to 1/10 the size of the original class files).

16 The Jalapeño dynamic optimizing compiler for Java

82%

- 4 Michael G. Burke , Jong-Deok Choi , Stephen Fink , David Grove , Michael Hind , Vivek Sarkar , Mauricio J. Serrano , V. C. Sreedhar , Harini Srinivasan , John Whaley
Proceedings of the ACM 1999 conference on Java Grande June 1999

17 Ajents: towards an environment for parallel, distributed and mobile Java applications 82%

- 4 Matthew Izatt , Patrick Chan , Tim Brecht
Proceedings of the ACM 1999 conference on Java Grande June 1999

18 Hypermedia systems: IUHM: a hypermedia-based model for integrating open services, data and 80%

- 4 metadata
Marc Nanard , Jocelyne Nanard , Peter King
Proceedings of the fourteenth ACM conference on Hypertext and hypermedia August 2003
This paper discusses a new hypermedia-based model known as IUHM (Information Unit Hypermedia Model). IUHM emerged as a result of the development of the OPALES system, a collaborative environment for exploring and indexing video archives in a digital library. A basic design requirement of OPALES is that it must permit and support the integration of new services throughout its life cycle. Thus, IUHM depends heavily upon the notions of extensibility and openness. Support for openness, extensibility ...

19 Ravenscar-Java: a high integrity profile for real-time Java 80%

- 4 Jagun Kwon , Andy Wellings , Steve King
Proceedings of the 2002 joint ACM-ISCOPE conference on Java Grande November 2002
For many, Java is the antithesis of a high integrity programming language. Its combination of object-oriented programming features, its automatic garbage collection, and its poor support for real-time multi-threading are all seen as particular impediments. The Real-Time Specification for Java has introduced many new features that help in the real-time domain. However, the expressive power of these features means that very complex programming models can be created, necessitating complexity in the ...

20 Session 2: OS architecture I: Multiprocessing and portability for PDAs 80%

- 4 Grzegorz Czajkowski
Proceedings of the 9th workshop on ACM SIGOPS European workshop: beyond the PC: new challenges for the operating system September 2000
The role of small devices in the emerging all-connected computer infrastructure is growing. So are the requirements that the application execution environments face. Portability, mobility, resource scarcity, and security concerns aggravated by often unknown sources of executed code combine together to create a challenging design and implementation task. It has been extensively argued and demonstrated that safe languages can solve some of these problems. In this paper, we focus on the multiproces ...

Results 1 - 20 of 60 [short listing](#)

This Page Blank (uspto)